REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-01-

0163

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	February 21, 2001	FINAL TECHNICAL - 5/1/97-10/31/99

4. TITLE AND SUBTITLE
Decomposing Large, Complex, Concurrent Systems into Manageable Building Blocks

5. FUNDING NUMBERS
F49620-97-1-0337

6. AUTHOR(S)
Nancy Lynch

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NM (Dr. Robert Herklotz)
801 No. Randolph, room 732
Arlington, VA 20332-8080

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
N/A

12a. DISTRIBUTION / AVAILABILITY STATEMENT
No limits of disclosure

12b. DISTRIBUTION CODE

This project produced many techniques to assist developers in decomposing large, complex, concurrent systems into manageable building blocks.

(1) (Foundations) A new mathematical modeling framework was developed for hybrid (discrete/continuous) systems. Also, new methods for using small finite-state abstractions in verifying complex distributed algorithms were developed.

(2) (Distributed systems) (2a) Compositional modeling and analysis methods were utilized in the design of several services for group-oriented computation in fault-prone distributed networks. In particular, group communication services supporting a changing set of participating clients, and a group membership service suitable for WANs, were designed and developed. The methods were also used to validate (and find an error in) the Ensemble group communication system. (2b) Substantial progress was made on a mathematical framework for analyzing weakly coherent multiprocessor memories and the programs that use them. (2c) The modeling and analysis methods were also used to develop a new weakly coherent ("eventually serializable") data service for distributed networks, and to analyze standard communication protocols, security protocols, and operating systems.

(3) (Automated transportation systems) The new hybrid system modeling framework was applied to the task of analyzing the behavior of automated multi-vehicle transportation systems. This work included analysis of both ground and air transportation systems.

(4) (Programming language development) The IOA specification and programming language was defined. A high-level design was produced for a toolset including validation and code generation tools, and substantial progress was made on detailed design and implementation of the toolset.

14. SUBJECT TERMS

15. NUMBER OF PAGES
34

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements.***

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD	- See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	- See authorities.
NASA	- See Handbook NHB 2200.2.
NTIS	- Leave blank.

Block 12b. Distribution Code.

DOD	- Leave blank.
DOE	- Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	- Leave blank.
NTIS	- Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

Contents

1 Project participants

Senior research staff:

Nancy Lynch, TDS Group Leader
Nir Shavit, Visiting Professor, Tel Aviv University
Alex Shvartsman, Postdoctoral Research Associate
John Lygeros, Postdoctoral Research Associate
Idit Keidar, Postdoctoral Research Associate

Visitors:

Alan Fekete, University of Sydney
Roberto Segala, University of Bologna
Sergio Rajsbaum, Universidad Nacional Autonoma de Mexico (UNAM)

Graduate Students:

Anna Chefter, MEng completed
Oleg Cheiner, MEng completed
Roberto De Prisco, MS completed, PhD completed
Ekaterina Dolginova, MEng completed
Gunnar Hoest, MS completed
Kyle Ingols, MEng completed
Henrik Jensen, PhD completed (Aalborg)
Roger Khazan, MS completed, PhD in progress
Carl Livadas, MS completed, PhD in progress
Victor Luchangco, PhD in progress
Antonio Ramirez, MEng completed
Mark Smith, PhD completed
Qixiang Sun, MEng completed
Igor Tarashchanskiy, MEng completed
Joshua Tauber, PhD in progress
Dan Touitou (Tel Aviv University)
Mandana Vaziri, MS completed

Undergraduate Students:

Tsvetomir Petrov
Holly Reimers
Michael Tsai
Daniel Yates (Harvard)

2 Overview

This project produced many techniques to assist developers in decomposing large, complex, concurrent systems into manageable building blocks. These techniques include:

1. General mathematical modeling frameworks and verification methods.
2. Specific building blocks for constructing complex systems, particularly in the area of fault-tolerant distributed computing, and specific analysis results for a wide variety of system designs.
3. Language and tool support for modeling, design, and analysis.

These are described in the following sections of this report. Those sections are organized generally according to the Statement of Work from the original proposal. For the reader's convenience, we reproduce that Statement here:

"We will develop mathematical models and techniques to support the decomposition of large, complex, concurrent systems into manageable building blocks. The systems to be considered include computers, software, communication mechanisms, physical processes, and human users. We consider only "true decomposition", in which each component comes equipped with a clear specification of what it accomplishes, so that the relevant behavior of the system as a whole may be understood entirely in terms of the component specifications. Specifications will include performance and fault-tolerance properties, as well as ordinary correctness properties. Such decomposition can be used for modular construction, description, specification, verification, maintenance, and reuse of systems.

Our work is based on a foundation consisting of several varieties of (not-necessarily-finite-state) automata, plus parallel composition and other operations for combining automata. Proof methods include invariant assertions, several forms of abstraction mappings, and compositional methods; all admit computer assistance.

Specifically:
(1) (Foundation) We have already completed the development of most of the foundation. We will attempt to complete the remaining foundational work, on automaton models for hybrid (discrete/continuous) systems, for probabilistic systems, and for systems with a combination of hybrid and probabilistic behavior.
(2) (Distributed Systems) We will provide careful specifications for the most important building blocks (e.g., memory components, communication and synchronization services) for efficient, reliable distributed systems. We will devise, analyze, and compare implementations of these building blocks, both in terms of other building blocks and in terms of basic distributed system models. We will seek related impossibility results. We will develop useful proof techniques for demonstrating the correctness, performance, and reliability properties of such implementations.
(3) (Automated Transportation Systems) We will model common scenarios arising in a typical hybrid system application: automated transportation systems (including both automated public transit and intelligent highway systems). We will attempt to prove the key safety, performance,

and reliability properties of those systems. In doing so, we will attempt to discover appropriate methods and building blocks for decomposing such systems, and useful proof, analysis, and comparison techniques. This work is intended to contribute a useful theory for this application domain, and also to assist in the development and validation of our basic models and methods.
(4) (Programming Language Development) As we gain experience in representing various types of systems and components, our representation techniques will become increasingly stylized. We will attempt to formalize these techniques as precisely-defined programming language constructs. Eventually, we intend to define a suite of programming languages appropriate for programming decomposable systems. These languages will be usable with several tools, including interactive theorem-provers, model-checkers, simulators, and compilers."

3 Foundations

Hybrid I/O Automata

Lynch, Segala, Vaandrager, and Weinberg completed a journal submission based on their *hybrid I/O automaton (HIOA)* mathematical framework [1] for hybrid systems. The framework supports modeling of both discrete events and system evolution over intervals of real time. It supports proofs using invariants, composition, and levels of abstraction.

We used the HIOA model to describe and analyze automated transportation system designs, including the designs for the TCAS II-7 aircraft collision avoidance system and the PATH intelligent highway system.

Carl Livadas, John Lygeros, and Nancy Lynch. High-level modeling and analysis of TCAS. In *Proceedings of the 20th Real-Time Systems Symposium (RTSS99*, Phoenix, Arizona, December 1999. `theory.lcs.mit.edu/tds/papers/Livadas/RTSS99.ps.gz`.

John Lygeros and Nancy Lynch. On the formal verification of the TCAS conflict resolution algorithms. In *36th IEEE Conference on Decision and Control*, pages 1829–1834, San Diego, CA, December 1997. Extended abstract.
`theory.lcs.mit.edu/tds/papers/Lygeros/CDC97.ps.gz`.

John Lygeros and Nancy Lynch. Conditions for safe deceleration of strings of vehicles. Research Report UCB-ITS-PRR-2000-2, California PATH Program, Institute of Transportation Studies, University of California, Berkeley, January 2000.

Ekaterina Dolginova and Nancy Lynch. Safety verification for automated platoon maneuvers: A case study. In Oded Maler, editor, *Hybrid and Real-Time Systems* (International Workshop, HART'97, Grenoble, France, March 1997), volume 1201 of *Lecture Notes in Computer Science*, pages 154–170. Springer-Verlag, 1997.
`theory.lcs.mit.edu/tds/papers/Dolginova/platoons.ps.gz`.

Constance Heitmeyer and Nancy Lynch. Formal verification of real-time systems using timed automata. In Constance Heitmeyer and Dino Mandrioli, editors, *Formal Methods for Real-Time Computing*, Trends in Software, chapter 4, pages 83–106. John Wiley & Sons Ltd, April 1996. `theory.lcs.mit.edu/tds/papers/Heitmeyer/FMRTC96.html`.

H. B. Weinberg, Nancy Lynch, and Norman Delisle. Verification of automated vehicle protection systems. In R. Alur, T. Henzinger, and E. Sontag, editors, *Hybrid Systems III: Verification and Control* (DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems, New Brunswick, New Jersey, October 1995), volume 1066 of *Lecture Notes in Computer Science*, pages 101–113. Springer-Verlag, 1996.
`theory.lcs.mit.edu/tds/papers/Weinberg/DIMACS95.ps.gz`.

Nancy Lynch. Modelling and verification of automated transit systems, using timed automata, invariants and simulations. In R. Alur, T. Henzinger, and E. Sontag, editors, *Hybrid Systems III: Verification and Control* (DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems, New Brunswick, New Jersey, October 1995), volume 1066 of *Lecture Notes in Computer Science*, pages 449–463. Springer-Verlag, 1996. Also, Technical Memo MIT/LCS/TM-545, Laboratory for Computer Science, Massachusetts Institute of Technology Cambridge, MA 02139, December 1995.
`theory.lcs.mit.edu/tds/papers/Lynch/transit-survey.html`.

Nancy Lynch and H. B. Weinberg. Proving correctness of a vehicle maneuver: Deceleration. In *Second European Workshop on Real-Time and Hybrid Systems*, pages 196–203, Grenoble, France, May/June 1995. Later version appears as [2].

H. B. Weinberg and Nancy Lynch. Correctness of vehicle control systems: A case study. In *17th IEEE Real-Time Systems Symposium*, pages 62–72, Washington, D. C., December 1996.
`theory.lcs.mit.edu/tds/papers/Weinberg/RTSS96.ps.gz`.

H. B. Weinberg. Correctness of vehicle control systems: A case study. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, February 1996. Also, MIT/LCS/TR-685.
`theory.lcs.mit.edu/tds/papers/Weinberg/SMThesis.html`.

Nancy Lynch, Roberto Segala, Frits Vaandrager, and H. B. Weinberg. Hybrid I/O automata. In R. Alur, T. Henzinger, and E. Sontag, editors, *Hybrid Systems III: Verification and Control* (DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems, New Brunswick, New Jersey, October 1995), volume 1066 of *Lecture Notes in Computer Science*, pages 496–510. Springer-Verlag, 1996.
`theory.lcs.mit.edu/tds/papers/Lynch/LSVW.html`.

Ekaterina Dolginova. Safety verification of automated car maneuvers. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, May 1998.
`theory.lcs.mit.edu/tds/papers/Dolginova/thesis.ps.gz`.

Michael S. Branicky, Ekaterina Dolginova, and Nancy Lynch. A toolbox for proving and maintaining hybrid specifications. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems IV (Fourth International Conference on Hybrid Systems, Ithaca, NY, October 1996)*, volume 1273 of *Lecture Notes in Computer Science*, pages 18–30. Springer Verlag, 1997. `theory.lcs.mit.edu/tds/papers/Branicky/BDL-toolbox.ps.gz`.

Carolos Livadas and Nancy A. Lynch. Formal verification of safety-critical hybrid systems. In S. Sastry and T.A. Henzinger, editors, *Hybrid Systems: Computation and Control* (First International Workshop, HSCC'98, Berkeley, CA, USA, April, 1998), number 1386 in Lecture Notes in Computer Science, pages 253–272. Springer Verlag, 1998. `theory.lcs.mit.edu/tds/Livadas/HS98.html`.

Recently, Lynch, Segala, and Vaandrager defined a simpler version of the model, which does not use shared variables for communication, but only shared discrete actions and continuous flows across component boundaries. A conference version has been produced, and a full version is still in progress.

Nancy Lynch, Roberto Segala, Frits Vaandrager, and H. B. Weinberg. Hybrid I/O automata. In *Hybrid Systems: Computation and Control: Fourth International Workshop (HSCC'01)*, Lecture Notes in Computer Science, Roma, Italy, March 2001. Springer-Verlag. To appear. `theory.lcs.mit.edu/tds/papers/Lynch/HIOA.ps`.

Small abstractions

Jensen developed techniques for integrating model checking and theorem proving methods for verification of concurrent systems. His methods involve defining property-preserving abstraction mappings from complex systems to small finite-state systems that can be model-checked. He formulated his methods in terms of I/O automata.

He applied his methods to parameterized distributed algorithms consisting of an unbounded number of identical components. In particular, he obtained a small finite-state abstraction of Burns' n-process mutual exclusion algorithm [3] The abstraction is safe with respect to the mutual exclusion property, i.e., if the abstraction satisfies mutual exclusion then so does the concrete parameterized system for any n. The abstraction can easily be verified to satisfy mutual exclusion, e.g., automatically by model-checking. Likewise, he proved the fundamental safety properties of the Bounded Concurrent Timestamp System (BCTSS) algorithm of Dolev and Shavit [4, 5] by constructing a property-preserving abstraction mapping to a model-checkable finite-state system.

The work on Burns' algorithm appears in:

Henrik Jensen and Nancy Lynch. A proof of Burns N-Process mutual exclusion algorithm using abstraction. In Bernhard Steffen, editor, *Tools and Algorithms for the Construction and Analysis of Systems* (TACAS'98, Lisbon, Portugal, April 1998), volume 1384 of *Lecture Notes in Computer Science*, pages 409–423. Springer, 1998. `theory.lcs.mit.edu/tds/papers/Jensen/burnspaper.ps.gz`.

A full description of the project appears in:

> Henrik Ejersbo Jensen. *Abstraction-Based Verification of Distributed Systems.* PhD thesis, Department of Computer Science, Institute for Electronic Systems, Aalborg University, Aalborg, Denmark, June 1999. R-99-5005.
> `theory.lcs.mit.edu/tds/papers/Jensen/Jensenthesis.ps.gz`.

Liveness properties

We completed our work on compositional liveness properties for discrete systems:

> Roberto Segala. Quiescence, fairness, testing and the notion of implementation. *Information and Computation*, 130(2):194–210, November 1997.
>
> Roberto Segala, Rainer Gawlick, Jørgen Søgaard-Andersen, and Nancy Lynch. Liveness in timed and untimed systems. *Information and Computation*, 141(2):119–171, March 1998.
> `theory.lcs.mit.edu/tds/papers/Segala/GLSS-liveness.ps.gz`.

Timing models and analysis methods

De Prisco and Lynch completed development of the Clock General Timed Automaton (Clock GTA) model, which provides a systematic way of describing discrete timing-based systems in which there is a notion of "normal" timing behavior, but that do not necessarily always exhibit this "normal" behavior. The Clock GTA model can be used for practical time performance analysis based on the stabilization of the physical system. It has been used to model, verify and analyze Lamport's Paxos algorithm for distributed consensus. This work appeared in:

> Roberto DePrisco. Revisiting the Paxos algorithm. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, June 1997. Also, MIT/LCS/TR-717.
> `theory.lcs.mit.edu/tds/papers/DePrisco/mastersthesis.ps.gz`.

Other foundations topics

We did not make significant progress on probabilistic system modeling during this contract. In particular, we did not reach the point of combining hybrid and probabilistic system modeling frameworks; this remains for future work, and seems like a hard problem.

4 Distributed Systems

We designed, modeled, and analyzed a large collection of building blocks for fault-tolerant distributed systems.

4.1 Distributed data management

Weakly coherent data

We completed a journal version of our earlier results on data services that provide a weak coherence condition known as *Eventual Serializability* [6].

This service supports tradeoffs between data coherence guarantees and performance.

> Alan Fekete, David Gupta, Victor Luchangco, Nancy Lynch, and Alex Shvartsman. Eventually-serializable data service. *Theoretical Computer Science*, 220(1):113–156, June 1999. Special Issue on Distributed Algorithms.
> **Abstract**: Data replication is used in distributed systems to improve availability, to increase throughput and to eliminate single points of failures. The disadvantage of replication is the additional effort required to maintain consistency among replicas. In some settings, e.g., distributed directory services, it is acceptable to have transient inconsistencies as long as a consistent view of data is eventually established through a well-specified serialization of operations. For such services to be usable, it is important that the consistency guarantees are specified clearly.
> We present a new specification for distributed data services that trades off immediate consistency guarantees for improved system availability and efficiency, while ensuring the long-term consistency of the data. An *eventually-serializable data service* maintains the requested operations in a partial order that gravitates over time towards a total order. It provides clear and unambiguous guarantees about the immediate and long-term behavior of the system. To demonstrate its utility, we present an algorithm, based on one of Ladin, Liskov, Shrira, and Ghemawat [7], that implements this specification. Our algorithm exports the interface of the abstract service, and generalizes their algorithm by allowing general operations and greater flexibility in specifying consistency requirements.
> `http://theory.lcs.mit.edu/~victor_l/eventually-serializable.html`.

The journal version contains several improvements over the conference version: explicit guarantees on the behaviors allowed by ESDS that may be helpful when ESDS is used as a building block in the design of applications; guarantees on the performance of the algorithm presented in the paper to implement ESDS; an analysis of the fault-tolerance of the algorithm under various timing and failure assumptions; and a more formal treatment of various optimizations.

Cheiner and Shvartsman implemented a prototype ESDS in a network of Unix workstations and performed empirical performance study of several prototype applications based on the service. The system is implemented in C++ and it uses MPI as the message passing mechanism. Performance tests using a network of 12 Sun workstations confirmed the expected scalability of ESDS in this setting. This work also confirmed that ESDS performance reflects a trade-off between data consistency and system responsiveness. This work appears in:

> Oleg Cheiner. Implementation and evaluation of an eventually-serializable data service. Masters of Engineering, MIT Department of Electrical Engineering and Computer Science.

Cambridge, MA, September 1997.
theory.lcs.mit.edu/tds/papers/Cheiner/MEngThesis.ps.gz.

Oleg Cheiner and Alex A. Shvartsman. Implementation of an eventually serializable data service. In *Proceedings of the 17th Annual ACM Symposium on the Principles of Distributed Computing*, Puerta Vallarta, Mexico, June 1998. (Short paper).

Oleg Cheiner and Alex Shvartsman. Implementing an eventually-serializable data service as a distributed system building block. In M. Mavronicolas, M. Merritt, and N. Shavit, editors, *Networks in Distributed Computing*, volume 45 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 43–72. American Mathematical Society, 1999.
Abstract: This work presents an implementation of a distributed system building block that is formally specified as the Eventually-Serializable Data Service (ESDS) proposed by Fekete *et al.* ESDS deals with replicated objects that allow the users of the service to relax consistency requirements in return for improved responsiveness, while providing guarantees of eventual consistency of the replicated data. ESDS includes a formal service specification and an abstract algorithm for it. The algorithm is given in terms of I/O automata of Lynch and Tuttle. An important consideration in formulating ESDS was that it could be employed in building real systems.
The work described here makes the following contributions. We develop an optimized implementation of ESDS and explore its behavior. We combine the implementation with different data types and clients, thus demonstrating the utility of the service as a *building block* suitable for serving as a distributed operating system component. The implementation has been experimentally evaluated on a network of workstations. The results confirm that the designed trade-off between *consistency* and *performance* is present in the implemented service. To make the implementation process less error prone, we develop and use a framework for mapping algorithms formally specified using I/O automata to distributed programs. The framework includes a set of conversion rules and a core set of object common to all target implementations.
theory.lcs.mit.edu/tds/papers/Cheiner/podcesds.ps.

Consensus

De Prisco, Lampson and Lynch completed their work on Lamport's apparently-practical Paxos algorithm for distributed consensus [8]. Using the Clock General Timed Automaton described above, they formally modeled, verified and analyzed the Paxos algorithm.

Roberto DePrisco. Revisiting the Paxos algorithm. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, June 1997. Also, MIT/LCS/TR-717.
theory.lcs.mit.edu/tds/papers/DePrisco/mastersthesis.ps.gz.

Roberto DePrisco, Butler Lampson, and Nancy Lynch. Revisiting the Paxos algorithm. In Marios Mavronicolas and Philippas Tsigas, editors, *Distributed Algorithms* 11th International

Workshop, WDAG'97, Saarbrücken, Germany, September 1997 Proceedings, volume 1320 of *Lecture Notes in Computer Science*, pages 111–125, Berlin-Heidelberg, 1997. Springer-Verlag. `theory.lcs.mit.edu/tds/DePrisco/wdag97-DLL97.ps.gz`.

Roberto DePrisco, Butler Lampson, and Nancy Lynch. Fundamental study: Revisiting the Paxos algorithm. *Theoretical Computer Science*, 243:35–91, 2000.

De Prisco worked with Drs. Dahlia Malkhi and Michael Reiter at AT&T Labs on distributed consensus building blocks in systems that tolerate stop failures and Byzantine failures. They studied several variations of the k-set consensus building block both in the message passing setting and in the shared memory model. For each variation of the problem the maximum fault tolerance of the system has been identified.

Roberto DePrisco, Dahlia Malkhi, and Michael K. Reiter. On k-set consensus problems in asynchronous systems. *IEEE Transactions on Parallel and Distributed Computing*, 12(1):7-21, January 2001.
`theory.lcs.mit.edu/tds/papers/DePrisco/tpds-110354.ps.gz`.
Earlier version appeared in *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing*, Atlanta, Georgia, May 1999.
`theory.lcs.mit.edu/tds/papers/DePrisco/podc99.ps.gz`.

Lynch and Segala completed both conference and journal versions of the following paper. The conference version appeared in honor of the memory of PhD student Anna Pogosyants, who was killed in a car accident in 1995.

Anna Pogosyants, Roberto Segala, and Nancy Lynch. Verification of the randomized consensus algorithm of Aspnes and Herlihy: a case study. In Marios Mavronicolas and Philippas Tsigas, editors, *Distributed Algorithms* 11th International Workshop, WDAG'97, Saarbrücken, Germany, September 1997 Proceedings, volume 1320 of *Lecture Notes in Computer Science*, pages 22–36, Berlin-Heidelberg, 1997. Springer-Verlag.
`theory.lcs.mit.edu/tds/papers/Pogosyants/WDAG97.html`

Anna Pogosyants, Roberto Segala, and Nancy Lynch. Verification of the randomized consensus algorithm of Aspnes and Herlihy: a case study. *Distributed Computing*, 13, July 2000. Also, Technical Memo MIT/LCS/TM-555, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, June 1997.
`theory.lcs.mit.edu/tds/papers/Pogosyants/WDAG97.html`

Strongly coherent data

The following two papers, based on work done prior to this contract, appeared in print during this contract. They are important to this project because they represent our first attempts to define building blocks to support coherent distributed data management. The first of these uncovered a bug in the Orca system and produced a proof of (an abstract version of) the repaired

system. The second presented a communication service that supports dynamic reconfiguration, while maintaining coherent data.

Alan Fekete, M. Frans Kaashoek, and Nancy Lynch. Implementing sequentially consistent shared objects using broadcast and point-to-point communication. *Journal of the ACM*, 45(1):35–69, January 1998. `theory.lcs.mit.edu/tds/papers/Fekete/orca-journal.ps`.

Nancy Lynch and Alex Shvartsman. Robust emulation of shared memory using dynamic quorum-acknowledged broadcasts. In *Twenty-Seventh Annual International Symposium on Fault-Tolerant Computing (FTCS'97)*, pages 272–281, Seattle, Washington, USA, June 1997. IEEE. `theory.lcs.mit.edu/tds/papers/Lynch/FTCS97.ps.gz`.

4.2 Group communication systems

Our major emphasis during this project was on group-oriented communication services—their properties, algorithms that implement them, and applications that use them.

View-Synchronous group communication service (VS)

Our first work in this area, contained in the first two papers below, provided a foundation and approach for all of our later work. Namely, Fekete, Lynch, and Shvartsman developed an automaton specification for a prototype group communication service that we called *VS* (for "view-synchrony"). Our service was based on, though not identical to, services used in the Isis [9] and Transis [10] systems. Like these other group communication services, VS maintains and informs clients about the current membership of the group (using uniquely identified "views"), and integrates multicast communication with the views.

We also developed an automaton description of an algorithm that uses VS to implement a Totally Ordered Broadcast service, based on earlier work by Keidar and Dolev [11] and by Amir et al. [12], in the Transis project [10]. We proved correctness and performance properties for the algorithm, using invariants, composition, and abstraction relations. Contributions of this work included not just the specific services and algorithms, but also our general mathematical approach to analyzing group communication systems.

Alan Fekete, Nancy Lynch, and Alex Shvartsman. Specifying and using a partitionable group communication service. In *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 53–62, Santa Barbara, CA, August 1997. Longer version in Technical Memo MIT-LCS-TM-570.
`theory.lcs.mit.edu/tds/papers/Fekete/PODC97.ps.gz`.

Alan Fekete, Nancy Lynch, and Alex Shvartsman. Specifying and using a partitionable group communication service. *ACM Transactions on Computer Systems*. To appear. Also, Technical Memo MIT-LCS-TM-570, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 02139, 1997.

Abstract: A new, simple formal specification is presented for a partitionable view-oriented group communication service. The specification consists of a state machine to express safety requirements and a timed trace property to express performance and fault-tolerance requirements. The specification is used to construct a totally-ordered-broadcast application, using an algorithm (based on algorithms of Amir, Dolev, Keidar and others) that reconciles information derived from different views of the group. Correctness of the resulting application is proved, and its performance and fault-tolerance analyzed. The specification has a simple implementation, based on a group membership algorithm of Cristian and Schmuck.
`theory.lcs.mit.edu/tds/papers/Lynch/TOCS.ps.gz`.

Load-balancing using group communication

We continued this work by developing computation task load-balancing algorithms based on VS and similar services:

Roger Khazan. Group communication as a base for a load-balancing, replicated data service. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, June 1998. The thesis presents, models, and verifies a distributed algorithm for a sequentially consistent replicated data service that load-balances queries and tolerates partitioning and merging of the network. The algorithm is constructed using the VSGC service of Fekete, Lynch, and Shvartsman, described above.
`theory.lcs.mit.edu/tds/papers/Khazan/msthesis.ps.gz`.

Roger Khazan, Alan Fekete, and Nancy Lynch. Multicast group communication as a base for a load-balancing replicated data service. In *12th International Symposium on Distributed Computing*, pages 258–272, Andros, Greece, September 1998.
Abstract: We give a rigorous account of an algorithm that provides sequentially consistent replicated data on top of the view synchronous group communication service previously specified by Fekete, Lynch and Shvartsman. The algorithm performs updates at all members of a majority view, but rotates the work of queries among the members to equalize the load. The algorithm is presented and verified using I/O automata.
`theory.lcs.mit.edu/tds/papers/Khazan/disc98_gc_lbrds.ps.gz`.

Shlomi Dolev, Roberto Segala, and Alex Shvartsman. Dynamic load balancing with group communication. In *6th International Colloquium on Structural Information and Communication Complexity (SIROCCO'99)*. Also, Technical Memo MIT/LCS/TM-588, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1998.
Abstract: This work considers the problem of efficiently performing a set of tasks using a network of processors in the setting where the network is subject to dynamic reconfigurations, including partitions and merges. A key challenge for this setting is the implementation of dynamic load balancing that reduces the number of tasks that are performed redundantly because of the reconfigurations. We explore new approaches for load balancing in dynamic networks that can be employed by applications using a group communication service. For

the n-processor, n-task load balancing problem defined in this work, the following specific results are obtained. (1) For the case of fully dynamic changes including fragmentation and merges we show that the termination time of any on-line task assignment algorithm is greater than the termination time of an off-line task assignment algorithm by a factor greater than *n/12*. (2) We present a load balancing algorithm that guarantees completion of all tasks in *all* fragments caused by partitions with work *O(n+fn)* in the presence of *f* fragmentation failures. (3) We develop an effective scheduling strategy for minimizing the task execution redundancy and we prove that our strategy provides each of the n processors with a schedule of *Theta* $(n^{1/3})$ tasks such that at most *one* task is performed redundantly by *any* two processors.

Some other researchers also continued by modeling and analyzing related algorithms in the same framework [13, 14]. Archer expanded our invariant proofs and checked them using the PVS theorem prover [15]; she used these proofs as a vehicle for developing user support for proving invariants using PVS.

Ensemble

After completing our work on the theoretical VS service, we used the same methods to model and analyze the actual Ensemble group communication system at Cornell [16, 17]. The Ensemble system has an interesting layered design, where successive layers introduce stronger ordering and reliability properties. Some work had been done already on formal verification for Ensemble [18], specifically, on code-level verification for local optimizations; however, this earlier work did not involve global service definitions.

In particular, we developed global specifications for two key layers— the "Virtual Synchrony" layer, and another layer that combines Virtual Synchrony with a consistent total ordering property— and we modeled the Ensemble algorithm that spans between these layers. We then attempted a proof, but instead discovered a significant logical error in a key "state exchange" sub-algorithm. After this was repaired in the actual system, we developed models and proofs for the repaired system.

Jason Hickey, Nancy Lynch, and Robbert van Renesse. Specifications and proofs for Ensemble layers. In Rance Cleaveland, editor, *Tools and Algorithms for the Construction and Analysis of Systems, Fifth International Conference, (TACAS'99), Amsterdam, the Netherlands, March 1999*, volume 1579 of *Lecture Notes in Computer Science*, pages 119–133. Springer-Verlag, 1999. Held as part of the Joint European Conferences on Theory and Practice of Software, ETAPS'99.

Abstract: Ensemble is a widely used group communication system that supports distributed programming by providing precise guarantees for synchronization, message ordering, and message delivery. Ensemble eases the task of distributed application programming, but as a result, ensuring the correctness of Ensemble itself is a difficult problem. In this paper we use I/O automata for formalizing, specifying, and verifying the Ensemble implementation. We focus specifically on message total ordering, a property that is commonly used to guarantee consistency within a process group. The systematic verification of this protocol led to the discovery of an error in the implementation.

`theory.lcs.mit.edu/tds/papers/Lynch/tacas99.ps`.

The same error was found to exist also in Ensemble's predecessor system, Horus. In continuing work at Cornell, Hickey developed incremental models and proofs for Ensemble algorithms, based on I/O automata and using the Nuprl system [18].

An interesting feature of our work on Ensemble is that the abstract I/O automaton code written for the Ensemble layer algorithm looks very similar to the actual ML code (written by Mark Hayden) that comprises the system implementation. We also gave a slightly lower-level IOA description for the algorithm, which appears to have a simple simulation mapping to the abstract version, and which appears to be "almost isomorphic" to the actual code.

Dynamic views

All the work described so far assumes a static universe of processes. We extended this work to develop group communication services for a more dynamic setting, in which the universe of processes may change over time.

Many applications with strong consistency requirements restrict their operation so that significant work is performed only in certain "primary" views. Primary views are designed so that only one can exist at a time, and so that information can always flow from one primary to the next. For example, designating any view with a majority of the universe of processes as primary guarantees the requirements. Using majority views as primaries may be reasonable if the universe of processes is static, but not if it changes over time. Yeger-Lotem et al. [19] designed an algorithm in which each primary is guaranteed to contain a majority of the previous primary, rather than a majority of the universe. (Their algorithm keeps track of a set of *possible* previous primaries and ensures intersection with all of them.)

Using ideas from [19], we developed an automaton specification for a new Dynamic View Service, DVS, which produces primary views and manages group communication within those views. DVS guarantees that views are delivered to clients in a consistent order, and that each new primary view intersects all possible previous primaries. We modeled and proved correctness of an algorithm that implements DVS, and of an algorithm that uses DVS to implement a consistent totally ordered broadcast application.

> Roberto DePrisco, Alan Fekete, Nancy Lynch, and Alex Shvartsman. A dynamic view-oriented group communication service. In *Proceedings of the 17th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 227–236, Puerto Vallarta, Mexico, June-July 1998. `theory.lcs.mit.edu/tds/papers/DePrisco/DVS.ps.gz`.
>
> Roberto De Prisco, Alan Fekete, Nancy Lynch, and Alex Shvartsman. A dynamic view-oriented group communication service. Full version. To appear as MIT Laboratory for Computer Science Technical Memo.

We also extended this work to allow views to contain extra structure useful for group-oriented applications, for example, a designated leader or sets of read quorums and write quorums. Such structure can be used, for example, to help achieve consistency and availability in the face of

transient changes in the set of processes. We call such augmented views "configurations". We developed automaton specifications for two kinds of global "dynamic configuration services", which allow configurations to change, provided that each configuration satisfies certain intersection properties with respect to the previous configuration. We also designed algorithms that implement our dynamic configuration services, by extending the algorithm from [19]. Finally, we developed two consistent replicated data algorithms that use our dynamic configuration services: a dynamic version of Lamport's Paxos algorithm [8] and a dynamic version of an algorithm of Attiya, et al. [20]. Both algorithms tolerate both long-term changes in the set of processes (using reconfiguration) and transient changes (using quorums). This work appears in:

> R. De Prisco, A. Fekete, N. Lynch, and A.A. Shvartsman. A dynamic primary configuration group communication service. In Prasad Jayanti, editor, *Distributed Computing* Proceedings of DISC'99 - 13th International Symposium on Distributed Computing, Bratislava, Slovak Republic, September 1999, volume 1693 of *Lecture Notes in Computer Science*, pages 64–78, Bratislava, Slovak Republic, 1999. Springer-Verlag-Heidelberg.
> **Abstract:** Quorum-based methods for managing replicated data are popular because they provide availability of both reads and writes in the presence of faulty behavior by some sites or communication links. We refer to a quorum system as a *configuration*. If a system lasts for a very long time, it may become necessary to alter the configuration, perhaps because some sites have failed permanently and others have joined the system, or perhaps because users want a different trade-off between read-availability and write-availability. There are subtle issues that arise in managing the change of quorums, including how to make sure that any operation using the new configuration is aware of all information from operations that used an old configuration, and how to allow concurrent attempts to alter the configuration.
> In this paper we use ideas from group management services, especially those providing a dynamic notion of primary view; with this we define an abstract specification of a system that presents each user with a consistent succession of configurations. The key contribution here is the intersection property, that determines how the new configurations must relate to previous ones. We demonstrate that our proposed specification is neither too strong nor too weak, by showing how it can be implemented, and also how it is sufficient for the correctness of a replicated data management algorithm running above it.
> `theory.lcs.mit.edu/tds/papers/Fekete/disc-final.ps`.

> Roberto DePrisco and Nancy Lynch. An algorithm for replicated atomic objects. Submitted for publication.
> `theory.lcs.mit.edu/tds/papers/DePrisco/dcl.ps`.

The following PhD thesis contains the dynamic view and dynamic configuration results, plus some of the consensus results described above.

> Roberto DePrisco. *On Building Blocks for Distributed Systems.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology. Cambridge, MA 02139, December 1999.

Abstract: This thesis investigates two building blocks for distributed systems: group communication services and distributed consensus services.
Using group communication services is a successful approach in developing fault tolerant distributed applications. Such services provide communication tools that greatly facilitate the development of applications. Though many existing systems are used in real world applications, there is still the need of providing formal specifications for the group communication services offered by these systems. Great efforts are being made by many researchers to provide such specifications. In this thesis we have tackled this problem and have provided specifications for group communication services. One of our specifications considers the notion of primary view; another one generalizes this notion to that of primary configurations (views with quorums). Both specifications are shown to be implementable. The usefulness of both specifications is demonstrated by applications running on top of them. Our specifications are tailored to dynamic systems, where processes join and leave the system even permanently. We also showed how the approach used to develop the specifications can be applied to transform known algorithms, designed for stating settings, in order to make them adaptable to dynamic systems.
Distributed consensus is the abstraction of many coordination problems, which are of fundamental importance in distributed systems. Distributed consensus has been thoroughly studied and one important result showed that it is not possible to solve consensus in asynchronous systems if failures are allowed. However in such systems it is possible to solve the k-set consensus problem, which is a relaxed version of the consensus problem: each participating process begins the protocol with an input value and by the end of the protocol it must decide on one value so that at most k total values are decided by all correct processes (the classical consensus problem requires that there be a unique value decided by all correct processes). In this thesis we have investigated the k-set consensus problem in asynchronous distributed systems. We extended previous work by exploring several variations of the problem definition and model, including for the first time investigation of Byzantine failures. We showed that the precise definition of the validity requirement, which characterizes what decision values are allowed as a function of the input values and whether failures occur, is crucial to the solvability of the problem. We introduced six validity conditions for this problem (all considered in various contexts in the literature), and we demarcated the line between possible and impossible for each case. In many cases this line is different from the one of the originally studied k-set consensus problem.
`theory.lcs.mit.edu/tds/papers/DePrisco/phdthesis.ps.gz`.

Group communication in WANs

Most group communication services have been designed to operate in local-area networks. Keidar and co-workers designed and implemented group communication services suitable for wide-area networks. The basis for these services is a scalable group membership service, implemented on a small set of membership servers:

I. Keidar, J. Sussman, K. Marzullo, and D. Dolev. A Client-Server Oriented Algorithm for

Virtually Synchronous Group Membership in WANs. In *20th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 356–365, Taipei, Taiwan, April 2000. Full version: MIT Technical Memorandum MIT-LCS-TM-593.

Abstract: We describe a novel scalable group membership algorithm designed for wide area networks (WANs). Our membership service does not evolve from existing LAN-oriented membership services; it was designed explicitly for WANs. Our algorithm provides agreement on membership in a single message round in most cases, yielding a low message overhead. It avoids flooding the network and uses a scalable failure detection service designed for WANs. Furthermore, our algorithm avoids notifying the application of obsolete membership views when the network is unstable, and yet it converges when the network stabilizes.

In contrast to most group membership services, we separate membership maintenance from reliable communication in multicast groups: membership is not maintained by every process, but only by dedicated servers. The membership servers are not involved in the communication among the members of the groups. This design makes our membership service scalable in the number of groups supported, in the number of members in each group, and in the topology spanned by each group. Our service is complemented by a virtually synchronous communication service which provides clients with full virtual synchrony semantics.

`theory.lcs.mit.edu/~idish/ftp/wan-memb-tr.ps`.

The new group communication algorithms for WANs appear in:

Idit Keidar and Roger Khazan. A client-server approach to virtually synchronous group multicast: Specifications and algorithms. In *20th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 344–355, Taipei, Taiwan, April 2000. Full version: MIT Lab. for Computer Science Technical Report MIT-LCS-TR-794.
`theory.lcs.mit.edu/tds/papers/Keidar/vs-tr.ps.gz`.

Igor Tarashchanskiy. Virtual synchrony semantics: Client-server implementation. Masters of Engineering, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, 2000.
`theory.lcs.mit.edu/tds/papers/Tarashchanskiy/igor-thesis.ps.gz`.

The new algorithms are efficient, for example, one of our algorithms (which provides Virtual Synchrony) needs only one round for state exchange, which can be done in parallel with the membership services agreement on views.

A by-product of our work on scalable group communication was our development of new methods of incremental modeling and proof for distributed system components modeled as I/O automata. In our scalable group communication work, we used these methods to develop models and (both safety and liveness) proofs. The new incremental methods are described in:

Idit Keidar, Roger Khazan, Nancy Lynch, and Alex Shvartsman. An inheritance-based technique for building simulation proofs incrementally. In *22nd International Conference on*

Software Engineering (ICSE), pages 478–487, Limerick, Ireland, June 2000.
Also, submitted for journal publication.

Abstract: This paper presents a formal design for a novel group multicast service that provides virtually synchronous semantics in asynchronous fault-prone environments. The design employs a client-server architecture in which group membership is maintained not by every process but only by dedicated membership servers while virtually synchronous group multicast is implemented by service end-points running at the clients. This architecture allows the service to be scalable in the topology it spans, in the number of groups, and in the number of clients. Our design allows the virtual synchrony algorithm to run in a single message exchange round, in parallel with the membership algorithm: it does not require pre-agreement upon a common identifier by the membership algorithm.

Specifically, the paper defines service semantics for the client-server interface, that is, for the group membership service. The paper then specifies virtually synchronous semantics for the new group multicast service, as a collection of safety and liveness properties. These properties have been previously suggested and have been shown to be useful for distributed applications. The paper then presents new algorithms that use the defined group membership service to implement the specified properties. The specifications and algorithms are presented incrementally, using a novel *inheritance*-based formal construct. The algorithm that provides the complete virtually synchronous semantics executes in a single message round, and is therefore more efficient than previously suggested algorithms providing such semantics. The algorithm has been implemented in C++. All the specifications and algorithms are presented using the I/O automaton formalism. Furthermore, the paper includes formal proofs showing that the algorithms meet their specifications. Safety properties are proven using invariant assertions and simulations. Liveness is proven using invariant assertions and careful operational arguments.

`theory.lcs.mit.edu/~idish/ftp/vs-icdcs.ps`.

Specifications

Roman Vitenberg, Idit Keidar, Gregory V. Chockler, and Danny Dolev. Group communication specifications: A comprehensive study. Technical Report MIT-LCS-TR-790, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, September 1999.

Abstract: View-oriented group communication is an important and widely used building block for many distributed applications. Much current research has been dedicated to specifying the semantics and services of *view-oriented Group Communication Systems (GCSs)*. However, the guarantees of different GCSs are formulated using varying terminologies and modeling techniques, and the specifications vary in their rigor. This makes it difficult to analyze and compare the different systems.

This paper provides a comprehensive set of clear and rigorous specifications, which may be combined to represent the guarantees of most existing GCSs. In the light of these specifications, over thirty published GCS specifications are surveyed. Thus, the specifications serve as

a unifying framework for the classification, analysis and comparison of group communication systems. The survey also discusses over a dozen different applications of group communication systems, shedding light on the usefulness of the presented specifications.
Defining meaningful GCSs is challenging; such systems typically run in asynchronous environments in which agreement problems that resemble the services provided by group communication services are not solvable. Therefore, many of the suggested specifications turned out to be too trivial, and in particular, solvable by weaker algorithms than the actual implementations. In this paper, the non-triviality issues are addressed by guaranteeing conditional liveness and by using external failure detectors. The resulting specifications are non-trivial on one hand, and allow implementation on the other.
This paper is aimed at both system builders and theoretical researchers. The specification framework presented in this paper will help builders of group communication systems understand and specify their service semantics; the extensive survey will allow them to compare their service to others. Application builders will find in this paper a guide to the services provided by a large variety of GCSs, which would help them chose the GCS appropriate for their needs. The formal framework may provide a basis for interesting theoretical work, for example, analyzing relative strengths of different properties and the costs of implementing them.
`theory.lcs.mit.edu/~idish/ftp/gcs-survey-tr.ps`.

Other group communication topics

Other related work includes our design of an Optimistic Virtual Synchrony service:

J. Sussman, I. Keidar, and K. Marzullo. Optimistic virtual synchrony. Technical Report MIT-LCS-TR-792, MIT Lab for Computer Science, November 1999. Also Technical Report CS1999-634 University of California, San Diego, Department of Computer Science and Engineering.
Abstract: Group communication systems are powerful building blocks that facilitate the development of fault-tolerant distributed applications. Such systems generally run in an asynchronous fault-prone environment, and provide semantics (called Virtual Synchrony) that mask the asynchrony and unreliability of the environment. In order to implement Virtual Synchrony semantics, group communication systems typically impose blocking periods during which applications are not allowed to send messages.
This paper presents a novel form of group communication, Optimistic Virtual Synchrony (OVS). OVS allows applications to send messages during periods in which existing group communication services block, by making optimistic assumptions on the network connectivity. OVS allows applications to determine the policy as to when messages sent optimistically should be delivered and when they should be discarded. Thus, OVS gives applications fine-grain control over the specific semantics they require, and does not impose costs for enforcing any semantics that they do not require. At the same time, OVS provides a single easy-to-use interface for all applications. The paper presents several examples of applications that may exploit OVS and empirical results that show the performance benefits of using OVS.
`theory.lcs.mit.edu/~idish/ftp/ovs-tr.ps`.

Finally, the following two papers describe algorithms from the Transis group communication system: a basic algorithm for totally ordered broadcast, and an application built upon the Transis group communication services:

> Idit Keidar and Danny Dolev. Totally ordered broadcast in the face of network partitions. *exploiting group communication for replication in partitionable networks.* In D. Avresky, editor, *Chapter 3 of Dependable Network Computing.* Kluwer Academic, 1999.
> **Abstract**: This chapter presents an algorithm for Totally Ordered Broadcast in the face of network partitions and process failures, using an underlying group communication service as a building block. The algorithm always allows a majority (or quorum) of connected processes in the network to make progress (i.e., to order messages), if they remain connected for sufficiently long, regardless of past failures. Furthermore, the algorithm always allows processes to initiate messages, even when they are not members of a majority component in the network. These messages are disseminated to other processes using a gossip mechanism. Thus, messages can eventually become totally ordered even if their initiator is never a member of a majority component. The algorithm guarantees that when a majority is connected, each message is ordered within at most two communication rounds, if no failures occur during these rounds.
> `theory.lcs.mit.edu/~idish/ftp/keidar-chapter2.ps`.

> T. Anker, D. Dolev, and I. Keidar. Fault tolerant video-on-demand services. In *19th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 244–252, June 1999. `theory.lcs.mit.edu/tds/papers/Keidar/icdcs99.ps.gz`.

4.3 Multiprocessor memories

Luchangco has developed a mathematical framework for analyzing weakly coherent multiprocessor memories and the programs that use them. The complete framework will appear in this PhD thesis, scheduled to be completed any day now:

> Victor Luchangco. *Memory Consistency Models for High Performance Distributed Computing.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, in progress.

The following two papers represent early progress on this framework. The first of these contains a framework for specifying and reasoning about precedence-based memory models, which can express any conditions that rely only on some operations being required to precede other operations. Within this framework, Luchangco has defined a generalized notion of sequential consistency, as well as a weaker consistency requirement called per-location sequential consistency, and has characterized conditions under which clients cannot distinguish the two types of memory. He has also proved that the algorithm used by the Cilk system [21] implements a per-location sequentially consistent memory. He has begun work to extend this framework to incorporate locks and memory barriers.

Victor Luchangco. Precedence based memory models. In Marios Mavronicolas and Philippas Tsigas, editors, *Distributed Algorithms* 11th International Workshop, WDAG'97, Saarbrücken, Germany, September 1997 Proceedings, volume 1320 of *Lecture Notes in Computer Science*, pages 215–229, Berlin-Heidelberg, 1997. Springer-Verlag.
`theory.lcs.mit.edu/~victor_1/papers/precedence.html`.

The second paper is on "computation-centric memory models", which characterize memories from the programmer's viewpoint. A computation is a generalization of parallel instruction streams, and the memory models are expressed in terms of completed computations, separating out the memory semantics from linguistic and scheduling issues. In terms of these models, Luchangco and Frigo defined a property called "constructibility", which characterizes models that can be implemented exactly by on-line algorithms. They also defined sequential consistency and per-location sequential consistency, along with a family of very weak consistency models called "dag consistency" models, which arose from work on the Cilk system. They showed the relationships among these models, and proved that per-location sequential consistency is the weakest constructible memory stronger than any dag consistent memory.

Matteo Frigo and Victor Luchangco. Computation-centric memory models. In *ACM Symposium on Parallel Algorithms and Architectures (SPAA'98)*, pages 240–249, Puerto Vallarta. Mexico, June-July 1998. `theory.lcs.mit.edu/~victor_1/computation.html`.

4.4 Other distributed systems topics

Compositional models and analysis results were given for a variety of other distributed systems.

Communication protocols

Mark Smith finished his PhD thesis entitled "Formal Verification of TCP and T/TCP" in September, 1997. The thesis contains a formal verification of TCP with unbounded counters and TCP with bounded counters. It also shows that the T/TCP protocol does not guarantee at-most-once semantics, but satisfies a weaker specification. The thesis also contains an impossibility result for the problem of reliable message delivery and conditionally-fast transactions. This problem is the one that the TCP/IP transport level protocol T/TCP is designed to solve. The impossibility result says that if the client and server do not have accurate clocks, then no protocol can solve this problem. In addition to the proof of impossibility, this work presents an interesting formal model for systems with local clocks that must have liveness requirements. This model is based on an earlier general model for timed systems with liveness requirements of Segala, Gawlick, Sogaard-Andersen and Lynch [22].

Mark Smith. Reliable message delivery and conditionally-fast transactions are not possible without accurate clocks. In *Proceedings of the 17th Annual ACM Symposium on the Principles of Distributed Computing*, pages 163–171, June 1998.
`www.research.att.com/~mass/papers/podc98.html`.

Mark Smith. *Formal Verification of TCP and T/TCP.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, September 1997.
`theory.lcs.mit.edu/tds/papers/Smith/phdthesis.ps.gz`.

Mark Smith. Formal verification of TCP. In *Proceedings of The Second Technical Conference on Telecommunications R&D in Massachusetts*, pages 279–299, Lowell, MA, March 1996.

Mark Smith. Formal verification of communication protocols. In Reinhard Gotzhein and Jan Bredereke, editors, *Formal Description Techniques IX: Theory, Applications, and Tools* FORTE/PSTV'96: Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols, and Protocol Specification, Testing, and Verification, Kaiserslautern, Germany, October 1996, pages 129–144. Chapman & Hall, 1996.
`theory.lcs.mit.edu/tdspapers/Smith/forte96.ps.gz`.

Working at IBM, MEng student Sun used our methods to model and analyze the design of an IBM reliable broadcast system. This work led to the discovery and repair of an algorithmic error. It also led to a clean, decomposed model that serves as system documentation.

Qixiang Sun. Reliable multicast for publish/subscribe systems. Master's of Engineering, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, May 2000.
`theory.lcs.mit.edu/tds/papers/Sun/QSthesis.ps`.

Security protocols

Lynch applied composition and abstraction methods based on I/O automata to the task of decomposing and verifying basic security protocols.

Nancy Lynch. I/O automaton models and proofs for shared-key communication systems. In *12th IEEE Computer Security Foundations Workshop (CSFW12)*, pages 14–29, Mordano, Italy, June 28-30 1999.
`theory.lcs.mit.edu/tds/papers/Lynch/CSFW.html`. Full version appears in MIT/LCS/TR-789, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, August 1999.
Abstract: The combination of two security protocols, a simple shared-key communication protocol and the Diffie-Hellman key distribution protocol, is modeled formally and proved correct. The modeling is based on the I/O automaton model for distributed algorithms, and the proofs are based on invariant assertions, simulation relations, and compositional reasoning. Arguments about the cryptosystems are handled separately from arguments about the protocols.
`theory.lcs.mit.edu/tds/papers/Lynch/Security-tr.html`.

Reliable disk storage

Vaziri developed a model and correctness proof for the main contoller algorithm for the RAID level 5 system.

> Mandana Vaziri and Nancy Lynch. Proving correctness of a controller algorithm for the RAID level 5 system. In *28th International Symposium on Fault-Tolerant Computing Systems (FTCS)*, Munich, Germany, June 1998. `theory.lcs.mit.edu/~vaziri/raid.html`. Also, full version exists as Masters Thesis and Technical Memo MIT/LCS/TM-576, Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139, December 1997.
> `theory.lcs.mit.edu/tds/papers/Vaziri/MV-thesis.ps.gz`.

Operating systems

Yates, Lynch, Luchangco, and Seltzer developed IOA models for the architecture of a standard operating system.

> Daniel Yates, Nancy Lynch, Victor Luchangco, and Margo Seltzer. I/o automaton model of operating system primitives, May 1999. Bachelors Thesis.
> **Abstract:** Current research in the field of operating systems has been very systems-oriented and result driven. Little theoretical research has been done in considering the formal ramifications of these systems level decisions, or in mapping out the topology of the standard operating system. Because of this, it is often difficult in operating systems work to get a clear picture of the high-level interactions between different OS services, or to apportion programming efforts across well-defined interfaces. Formal specification of the operating system structure and methodology would provide a framework for clearer study, discussion, and implementation of operating systems.
> We present a formal method for modelling an operating system as a distributed system of state machines. Drawing the connection between the various independent services of an oeprating system and the independent agents of distributed systems, we model each service of the operating system as an asynchronous I/O Automaton. Demonstrating both the instructional and functional value of this modelling technique, we present here two views of the operating system. The first view, the User Level model, provides a simplified abstraction of the operating system. This acts as a operating system interface specification as well as an easy first step for teaching students in the field. The second view, or Kernel Level model, provides an implementation of the User Level abstract specification, and unveils many of the realities which were abstracted away in the User Level model. It provides a framework for research in formalizing operating systems, as well as providing a clear and concise description of many of today's operating system elements. Finally, using the powerful mathematical tools developed for I/O Automata, we make two formal modifications to the Kernel Level model and prove that it in fact does implement the User Level specification. We thereby assert that the two models are, from the perspective of the processes, functionally equivalent.
> `theory.lcs.mit.edu/tds/papers/Yates/danyates-thesis.ps`.

5 Automated Transportation Systems

The new hybrid system modeling framework was applied to the task of analyzing the behavior of automated multi-vehicle transportation systems. This work included analysis of both ground and air transportation systems.

Automated vehicle control systems

The following papers contains models and analyses, based on hybrid I/O automata, of simple vehicle maneuvers arising in people-mover designs. In particular, we analyzed vehicle protection subsystems, as used in the Raytheon Personal Rapid Transit project (PRT 2000tm). Our model allows simple composition of protectors that depend on each other's correct operation. The correctness proofs of the various protectors are facilitated by the proof of correctness of an abstract protector — a generic protector that captures the abstract functionality of a protector without considering the particular physical plant and protector details. The protectors modelled guard against overspeed and collision on a general track topology involving Y-shaped merges and diverges.
`theory.lcs.mit.edu/~clivadas/research.html`.

Carolos Livadas. Formal verification of safety-critical hybrid systems. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, September 1997. Also, MIT/LCS/TR-730, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, September 1997.
`theory.lcs.mit.edu/tds/papers/Livadas/MEngThesis.ps.gz`.

Carolos Livadas and Nancy A. Lynch. Formal verification of safety-critical hybrid systems. In S. Sastry and T.A. Henzinger, editors, *Hybrid Systems: Computation and Control* (First International Workshop, HSCC'98, Berkeley, CA, USA, April, 1998), number 1386 in Lecture Notes in Computer Science, pages 253–272. Springer Verlag, 1998.
`theory.lcs.mit.edu/tds/Livadas/HS98.html`.

Nancy Lynch. A three-level analysis of a simple acceleration maneuver, with uncertainties. In *AMAST Workshops on Real-Time Systems 95 & 96*, AMAST Series in Computing. World Scientific Publishing Company. To appear.
`theory.lcs.mit.edu/tds/papers/Lynch/three-level-paper.html`.

Platoons of cars

The following papers contain analyses of safety properties of vehicle platoon systems; the platoon design was considered by the State of California for use on some of its highways. Again, hybrid I/O automata were used. Specifically, we considered the problem of emergency deceleration of strings of vehicles. This problem involves hybrid dynamics, due to the interaction between the deceleration of the vehicles (continuous) and possible collisions (discrete). The project aimed to establish conditions under which such a maneuver can be executed safely, i.e., in a way that, if

any collisions occur, the relative velocity at impact is guaranteed to be low. For this purpose, we developed a model to capture the possibly complicated interaction between the continuous and discrete dynamics. Based on this model, we derived necessary and sufficient safety conditions for a particular deceleration maneuver, one where all vehicles brake as hard as possible until they come to a stop. The implications of these conditions for the safety of an AHS architecture that supports platooning of vehicles were also investigated.
`theory.lcs.mit.edu/tds/platoons.html`.

Ekaterina Dolginova. Safety verification of automated car maneuvers. Masters of Engineering, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA 02139, May 1998.
`theory.lcs.mit.edu/tds/papers/Dolginova/thesis.ps.gz`.

John Lygeros and Nancy Lynch. Strings of vehicles: Modeling and safety conditions. In S. Sastry and T.A. Henzinger, editors, *Hybrid Systems: Computation and Control* (First International Workshop, HSCC'98, Berkeley, CA, USA, April, 1998), number 1386 in Lecture Notes in Computer Science, pages 273–288. Springer Verlag, 1998.
`theory.lcs.mit.edu/tds/papers/Lygeros/strings.ps`.

Aircraft systems

The next two papers contain an abstract model for the Traffic Alert and Collision Avoidance System (TCAS) – a system used by commercial aircraft so as to assist pilots in resolving conflicts with other aircraft and preventing midair aircraft collisions. The modeling approach involves specifying all components of the system, in this case, not only the system hardware and software, but also the aircraft and pilots.

The second paper also contains analysis of some of the TCAS system's safety guarantees. A simplified version of a TCAS system is considered, involving two aircraft only, involving no delays or uncertainties, and in which pilots that obey the TCAS advisories. A characterization is given for situations in which collisions are successfully resolved by the TCAS system. A preliminary version of a basic safety (collision avoidance) analysis was presented to the Lincoln Labs TCAS group in December.

John Lygeros and Nancy Lynch. On the formal verification of the TCAS conflict resolution algorithms. In *36th IEEE Conference on Decision and Control*, pages 1829–1834, San Diego, CA, December 1997. Extended abstract.
`theory.lcs.mit.edu/tds/papers/Lygeros/CDC97.ps.gz`.

Carl Livadas, John Lygeros, and Nancy Lynch. High-level modeling and analysis of TCAS. In *Proceedings of the 20th Real-Time Systems Symposium (RTSS99*, Phoenix, Arizona, December 1999.
`theory.lcs.mit.edu/tds/papers/Livadas/RTSS99.ps.gz`.

In a similar project, Lygeros worked on formally modeling and proving the safety of the CTAS software. CTAS is designed to assist air traffic controllers with scheduling and routing of aircraft.

George Pappas, Shankar Sastry, John Lygeros, and Nancy Lynch. Modeling, specification, and safety analysis of CTAS. Final report, University of California, September 30 1997. Prepared for Langley Research Center and NEXTOR: FAA Center of Excellence in Aviation Operations Research.

George Pappas, Claire Tomlin, John Lygeros, Datta Godbole, and Shankar Sastry. A next generation architecture for air traffic management systems. In *36th IEEE Conference on Decision and Control*, San Diego, CA, December 1997. Extended abstract.

Darren Cofer, John Maloney, Rosa Weber, George Pappas, Shankar Sastry, John Lygeros, and Nancy Lynch. Formal specification and analysis of the center-TRACON automation systems (CTAS). Final Report SST-C97-002, Honeywell Technology Center, September 30 1997. Prepared for Langley Research Center and NEXTOR: FAA Center of Excellence in Aviation Operations Research.

6 Programming Language Development

In order to support modeling and analysis using I/O automata, we have defined a new language, IOA, for describing (discrete, untimed) specifications and algorithms. IOA is based on the informal notation used in Lynch's Distributed Algorithms book. This language is intended to be used for verification and analysis, using a variety of methods, and also as a basis for generation of running code. The following is a language manual for IOA:

Stephen J. Garland, Nancy A. Lynch, and Mandana Vaziri. IOA: A language for specifying, programming and validating distributed systems, December 2000. User and Reference Manual. `theory.lcs.mit.edu/tds/papers/Garland/ioaManual.ps.gz`.

The following two papers describe the high-level design of the IOA language and toolset. The toolset is intended to support distributed system development, using composition and levels of abstractions, from high-level specifications to efficient running code. Included in these papers is a formal description of a sample distributed banking application, together with invariants and simulation relations that can be used to prove its correctness. The proof has been done formally using the Larch Prover.

Stephen J. Garland and Nancy A. Lynch. The IOA language and toolset: Support for designing, analyzing, and building distributed systems. Technical Report MIT/LCS/TR-762, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, August 1998. `theory.lcs.mit.edu/tds/papers/Lynch/IOA-TR-762.ps`.

Stephen J. Garland and Nancy A. Lynch. *Using I/O Automata for Developing Distributed Systems*, chapter 13, pages 285–312. Cambridge University Press, USA, 2000. **Abstract:** This paper describes a new language for distributed programming, *IOA*, and high-level designs for a set of tools to support *IOA* programming. Both the language and the tools are based formally on the mathematical *I/O automaton model* for reactive systems. The language supports structured modeling of distributed systems using shared-action composition and levels of abstraction. The tools are intended to support design, several kinds of analysis, and generation of efficient runnable code.
`theory.lcs.mit.edu/tds/papers/Lynch/ioa-leavens.ps`.

Garland, with help from Chefter and other students, has written a front end for the language (parser and static-semantic checker); a revised version of the front end is now nearing completion. Chefter, Garland, and Lynch also developed a design and preliminary implementation for a "composer" program to transform IOA programs described using composition into primitive form.

Chefter, Garland, and Lynch developed a detailed design for a simulator for IOA, and Chefter produced a preliminary implementation:

Anna E. Chefter. A simulator for the IOA language. Masters of Engineering, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1998.
`theory.lcs.mit.edu/tds/papers/Chefter/thesis.ps.gz`.

Ramirez continued this work by designing and building a more advanced IOA simulator; this newer simulator allows *paired simulation* of two automata at once, with a check that one implements the other in the simulated execution.

J. Antonio Ramírez-Robredo. Paired simulation of I/O automata. Masters of Engineering, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, September 2000.
`theory.lcs.mit.edu/tds/papers/Ramirez/Ramirez-thesis.ps.gz`.

Ongoing work includes a code generator, which will generate running distributed code from IOA distributed algorithm descriptions. Tauber produced a preliminary (somewhat ad hoc) code generator, and Tauber and Tsai are now working on producing a complete implementation. First, Tauber defined the interface between IOA and a communication service that node IOA programs may use to communicate. He selected the Message Passing Interface (MPI) for this purpose. He designed a set of I/O automata that formally describe the behavior of an implemented MPI system and proved that these automata exhibit the desired reliable communication characteristics. Second, he devised a translation scheme from IOA programs to Java programs, and implemented a translator for a subset of these programs based on this scheme, using the existing parser. Translated programs are attached to Java libraries that expose external system services (e.g., console I/O and network communication) and implement IOA data types. Initially, scheduling nondeterminism is resolved

by a round-robin dispatcher. The next step is to generalize the translator to support higher-level communication abstractions, programmer defined data types, and a variety of methods for resolving nondeterminism.
`theory.lcs.mit.edu/~josh/codegen.html`.

Connections with verification tools are in earlier stages of design and development. A connection with the Larch Prover is planned. Vaziri designed and implemented a preliminary version of an interface to the SPIN model-checker. See `theory.lcs.mit.edu/tds/mc.html`. Two technical reports detailing example IOA applications are also in progress.

7 Other contributions

A variety of other projects don't quite fit into the main categories above. For completeness, we list them here.

Counting networks

Lynch, Shavit, Shvartsman, and Touitou studied the popular *counting network* data structure [23]. They discovered a precise characterization of the timing conditions under which such networks exhibit the strong data coherence condition of *linearizability*. They used these analytical results to explain the observations obtained for a simulated multiprocessor system.

> Nancy Lynch, Nir Shavit, Alex Shvartsman, and Dan Touitou. Timing conditions for linearizability in uniform counting networks. *Theoretical Computer Science*, 220(1):67–91, June 1999. Special Issue on Distributed Algorithms.

Parallel processing

Chlebus, De Prisco, and Shvartsman devised a new algorithm for executing tasks in a message-passing network of processors subject to processors' failures and restarts. Previous algorithms considered the same problem but only with failures, no restarts.

> B.S. Chlebus, R. De Prisco and A.A. Shvartsman. Performing Tasks on Synchronous Restartable Message-Passing Processors. *Distributed Computing*, 14(1): 49-64, 2001.
>
> Bogdan Chlebus, Roberto DePrisco, and Alex Shvartsman. Performing tasks on restartable message-passing processors. In Marios Mavronicolas and Philippas Tsigas, editors, *Distributed Algorithms* 11th International Workshop, WDAG'97, Saarbrücken, Germany, September 1997 Proceedings, volume 120 of *Lecture Notes in Computer Science*, pages 96–110 Berlin-Heidelberg, 1997. Springer-Verlag.
> `theory.lcs.mit.edu/tds/doall.html`.

The following book was written, giving basic results about costs of parallel computing in the presence of failures.

Paris C. Kanellakis and Alex A. Shvartsman. *A Theory of Fault-Tolerant Parallel Computation.* Kluwer Academic Publishers, Boston, MA, 1997.

Topology and complexity

Hoest and Shavit developed a notion of complexity for fault-tolerant asynchronous systems. They used topological models and methods to analyze time complexity in the "non-uniform iterated immediate snapshot" model, a restricted type of atomic snapshot shared memory model. They obtained tight bounds for the approximate agreement problem, and a fundamental "time vs. number of names" tradeoff for the processor renaming problem.

Gunnar Hoest. Towards a topological characterization of complexity in asynchronous, distributed systems. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, September 1997.
`theory.lcs.mit.edu/tds/papers/Hoest/Hoest-thesis.ps.gz`.

Gunnar Hoest and Nir Shavit. Towards a Topological Characterization of Asynchronous Complexity. *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 199-208, Santa Barbara, CA, August, 1997.
`theory.lcs.mit.edu/tds/papers/Hoest/HS-podc_final.ps`.

Distributed simulation

Lynch and Rajsbaum worked with Borowsky and Gafni to produce a model and careful proof for the important Borowsky-Gafni simulation result for fault-tolerant distributed systems.

Elizabeth Borowsky, Eli Gafni, Nancy Lynch, and Sergio Rajsbaum. The BG distributed simulation algorithm. to appear in *Distributed Computing.* Also, Technical Memo MIT/LCS/TM-573, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, December 1997.
`theory.lcs.mit.edu/tds/papers/Borowsky/TM-573.html`.

8 Technology transfer

Specific contact information for the following items is available upon request.

1. Lynch's book, "Distributed Algorithms", unifies a field that previously was described only in research papers. The book places the field on a mathematical foundation, and presents all the basic results and a selection of advanced topics. Particular features include an organization based on the timing model, and a compositional treatment of fault-tolerant systems. The book is aimed at graduate students and industrial researchers. It has been used as a textbook in many universities, and for summer courses for industrial computer system developers. It is now in its fourth printing.

2. Many researchers have used, or are using, our I/O automaton framework for their own work. For example, Gibbons, Merritt, and Gharachorloo [24] have modeled and verified properties of weakly coherent memories using I/O automata; Neiger at Intel has used them for modeling and verifying some properties of the Pentium family of processors; members of the Transis group at Hebrew University have used I/O automata to model their group communication systems [14]; Chaudhuri has used it to obtain results about fault-tolerant distributed computability; Castro and Liskov are using it to model/verify a Byzantine fault-tolerant distributed file system design [25]; and Lampson's Principles of Computer Systems graduate course uses our models and some of our proof methods as a foundation for its presentation of important ideas of software systems.

3. Our work on proofs for distributed algorithms has led to various improvements and additions to the Larch theorem proving system [26]. Also, Archer and Heitmeyer of the Naval Research Lab have used many of our hand-generated proofs as examples for their work on interactive theorem-proving using PVS; this work has had some indirect impact on improvements to PVS. Our examples have also been used as case studies by researchers using Isabelle and NuPRL.

4. Smith's work on formal modelling, verification and analysis of T/TCP [27] (an optimization of TCP) has provided protocol designers with new information about the limitations of what their protocol achieves.

5. Lynch and Fekete worked with Kaashoek on modelling the Orca shared object language and Amoeba operating system. Their work uncovered a bug in the Orca system and produced a proof of (an abstract version of) the repaired system.

6. Our work on view-synchronous group communication has been very well received in the community of systems researchers who design and build such systems, including the Transis group at Hebrew University, the Ensemble group at Cornell, and the (former) Isis group at Stratus Computer. Active research collaborations with the Transis and Ensemble groups have evolved. In particular, Lynch's visit to Cornell during Spring, 1998 led to models and proofs for key portions of the Ensemble system, plus the discovery of key errors in the Ensemble and Horus systems. Our models have also been used by several members of the Transis group, and Dr. Idit Keidar (of Transis) joined our group as a postdoc. Ongoing discussions with many people in this systems community are continually suggesting directions for our research, and are contributing insights to the systems researchers about their designs.

7. Vaziri's work on modelling/verifying RAID disk systems has helped to clarify work of Courtright and Gibson at CMU. For example, it has uncovered an error in the RAID level 6 design, and also another situation where more constraints on concurrency than necessary were used.

8. Assertional methods for timed I/O automata have been used by Vaandrager in projects to verify consumer electronics products at Philips electronics corporation.

9. Our work on modeling automated transportation systems led to a close collaboration with members of the Berkeley PATH automated highway project. In particular, John Lygeros, a control theorist working on the PATH project joined our group as a postdoc for a year, helping to apply a combination of our methods and those of control theory to several transportation case studies.

10. Our results about TCAS were reported back to engineers at Lincoln Labs and TASC, who originally started us working on this problem. We hope that our methods can be used as part of future efforts to validate similar aircraft control protocols.

11. Our work on weakly coherent shared memory resulted in collaborations with members of Leiserson's CILK project. In particular, we helped to clarify relationships among various specifications of such memory, including their "dag consistency" correctness condition.

12. Our work on IOA has created considerable interest among other researchers and developers working on applied formal methods. For example, Marco Devillers at U. Nijmegen is writing an interface from IOA to PVS, and members of the NuPRL group at Cornell have been formalizing IOA using NuPRL.

References

[1] Nancy Lynch, Roberto Segala, Frits Vaandrager, and H. B. Weinberg. Hybrid I/O automata. In R. Alur, T. Henzinger, and E. Sontag, editors, *Hybrid Systems III: Verification and Control* (DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems, New Brunswick, New Jersey, October 1995), volume 1066 of *Lecture Notes in Computer Science*, pages 496–510. Springer-Verlag, 1996. Also, submitted for journal publication and [28].

[2] H. B. Weinberg and Nancy Lynch. Correctness of vehicle control systems: A case study. In *17th IEEE Real-Time Systems Symposium*, pages 62–72, Washington, D. C., December 1996. Earlier version appears as [29].

[3] James E. Burns and Nancy A. Lynch. Bounds on shared memory for mutual exclusion. *Information and Computation*, 107(2):171–184, December 1993. Earlier conference version in [30].

[4] Danny Dolev and Nir Shavit. Bounded concurrent time-stamp systems are constructible. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, pages 454–466, Seattle, Washington, May 1989. Also, [31]. Later version in [32].

[5] Rainer Gawlick, Nancy Lynch, and Nir Shavit. Concurrent time-stamping made simple. In D. Dolev, Z. Galil, and M. Rodeh, editors, *Theory of Computing and Systems* (ISTCS'92, Israel Symposium, Haifa, Israel, May 1992), volume 601 of *Lecture Notes in Computer Science*, pages 171–185. Springer-Verlag, 1992.

[6] Alan Fekete, David Gupta, Victor Luchangco, Nancy Lynch, and Alex Shvartsman. Eventually-serializable data services. In *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 300–309, Philadelphia, PA, May 1996.

[7] Rivka Ladin, Barbara Liskov, Liuba Shrira, and Sanjay Ghemawat. Providing high availability using lazy replication. *ACM Transactions on Computer Science*, 10(4):360–391, 1992.

[8] L. Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, May 1998. Also Research Report 49, Digital Equipment Corporation Systems Research Center, Palo Alto, CA, September 1989.

[9] K. P. Birman and R. van Renesse. *Reliable Distributed Computing with the Isis Toolkit.* IEEE Computer Society Press, Los Alamitos, CA, 1994.

[10] D. Dolev and D. Malki. The Transis approach to high availability cluster communications. *Communications of the ACM*, 39(4):64–70, 1996.

[11] I. Keidar and D. Dolev. Efficient message ordering in dynamic networks. In *15th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 68–76, May 1996.

[12] Y. Amir, D. Dolev, P. Melliar-Smith, and L. Moser. Robust and efficient replication using group communication. Technical Report CS94-20, Institute of Computer Science, Hebrew University, Jerusalem, Israel, 1994.

[13] N. Lesley and A. Fekete. Providing view synchrony for group communication services. In *Proceedings of the Australian Computer Science Conference*, pages 457–468, Auckland, New Zealand, January 1999.

[14] Gregory V. Chockler. An adaptive totally ordered multicast protocol that tolerates partitions. Master's thesis, Institute of Computer Science, The Hebrew University of Jerusalem, Jerusalem, Israel, August 1997.

[15] Myla Archer. Personal communication.

[16] Robbert van Renesse, Ken Birman, Mark Hayden, Alexey Vaysburd, and David Karr. Building adaptive systems using Ensemble. *Software-Practice and Experience*, 29(9):963–979, July 1998.

[17] Mark G. Hayden. *The Ensemble System.* PhD thesis, Department of Computer Science, Cornell University, January 1997.

[18] Mark Bickford and Jason Hickey. Composition and inheritance for I/O automata using intersection types. Manuscript, 1998.

[19] E. Yeger Lotem, I. Keidar, and D. Dolev. Dynamic voting for consistent primary components. In *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 63–71, Santa Barbara, CA, August 1997.

[20] Hagit Attiya, Amotz Bar-Noy, and Danny Dolev. Sharing memory robustly in message-passing systems. *Journal of the ACM*, 42(1):124–142, January 1995.

[21] Robert D. Blumofe, Christopher F. Joerg, Bradley C. Kuszmaul, Charles E. Leiserson, Keith H. Randall, and Yuli Zhou. Cilk: An efficient multithreaded runtime system. In *Proceedings of the Fifth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, pages 207–216, Santa Barbara, California, July 1995.

[22] Roberto Segala, Rainer Gawlick, Jørgen Søgaard-Andersen, and Nancy Lynch. Liveness in timed and untimed systems. *Information and Computation*, 141(2):119–171, March 1998. composition/GLSS-liveness.ps.gz.

[23] James Aspnes, Maurice Herlihy, and Nir Shavit. Counting networks. *Journal of the ACM*, 41(5):1020–1048, September 1994. Also, Earlier version in *Proceedings of the 23rd ACM Annual Symposium on Theory of Computing*, pp. 348–358, May 1991. Also, MIT Technical Report MIT/LCS/TM-451, June 1991.

[24] Phillip B. Gibbons, Michael Merritt, and Kourosh Gharachorloo. Proving sequential consistency of high-performance shared memories. In *Symposium on Parallel Algorithms and Architectures*, pages 292–303, July 1991. Also AT&T Bell Laboratories Technical Report. May, 1991. (Submitted for publication.).

[25] Miguel Castro. *Practical Byzantine Fault Tolerance.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, November 2000.

[26] Stephen J. Garland and John V. Guttag. LP, the Larch Prover. `www.sds.lcs.mit.edu/~garland/LP/ove`

[27] Mark Smith. *Formal Verification of TCP and T/TCP.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, September 1997.

[28] Nancy Lynch, Roberto Segala, Frits Vaandrager, and H. B. Weinberg. Hybrid I/O automata. Technical Memo MIT/LCS/TM-544, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, December 1995. Also, [1].

[29] Nancy Lynch and H. B. Weinberg. Proving correctness of a vehicle maneuver: Deceleration. In *Second European Workshop on Real-Time and Hybrid Systems*, pages 196–203, Grenoble, France, May/June 1995. Later version appears as [2].

[30] James Burns and Nancy A. Lynch. Mutual exclusion using indivisible reads and writes. In *Proceedings of the 18th Allerton Conference on Communication, Control and Computing*, pages 833–842, Monticello, IL, October 1980.

[31] Danny Dolev and Nir Shavit. Bounded concurrent time-stamp systems are constructible. Technical Memo MIT/LCS/TM-393, June 1989.

[32] Danny Dolev and Nir Shavit. Bounded concurrent time stamping. *SIAM Journal on Computing*, 26(2):418–455, April 1997.